

Forschungszentrum Telekommunikation Wien

[Telecommunications Research Center Vienna]

Theory and Design of Turbo and Related Codes

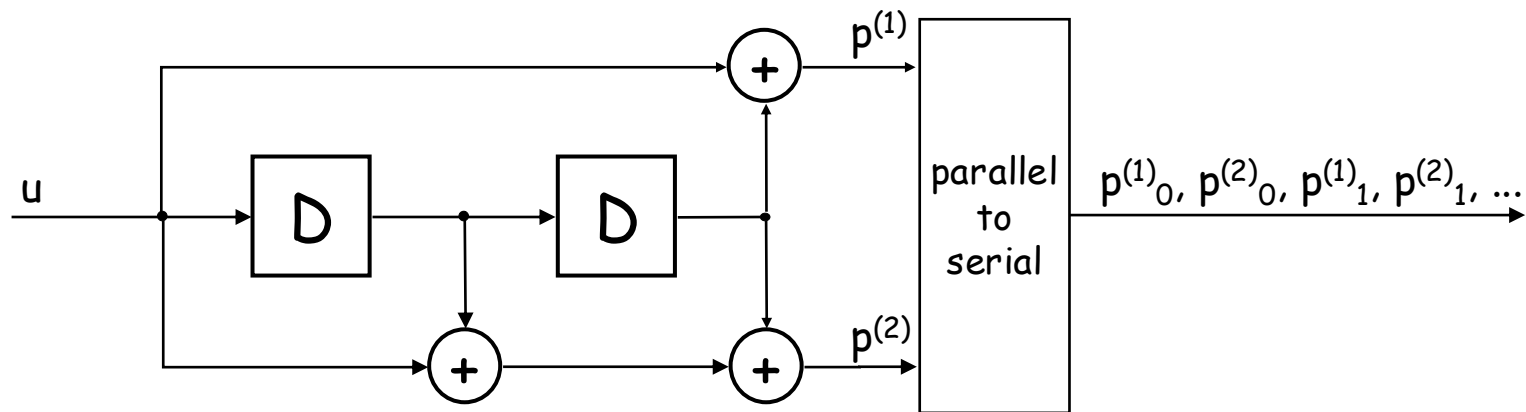
Lecture 7

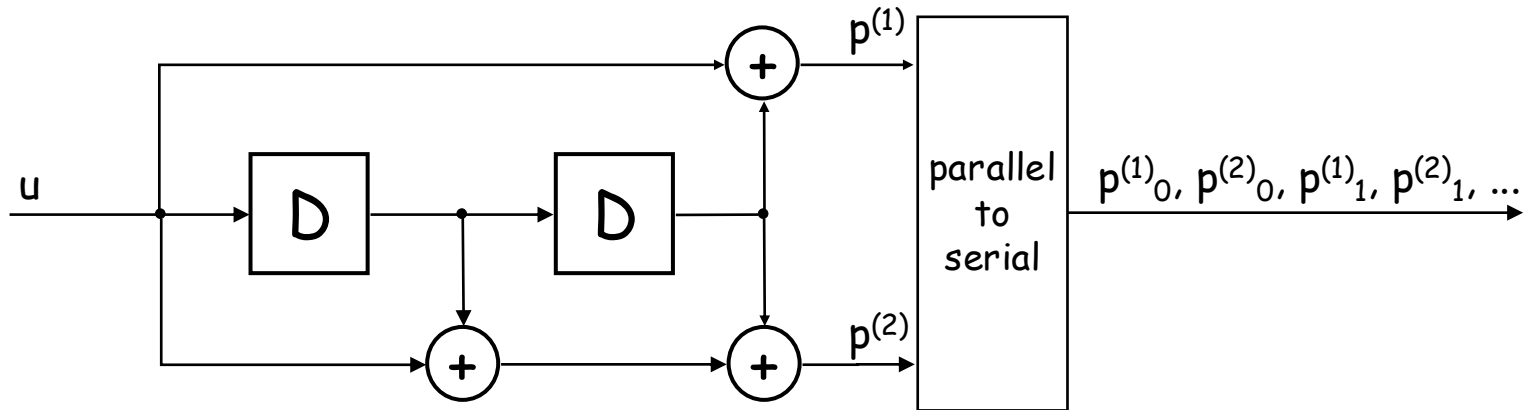
Jossy Sayir & Gottfried Lechner

<http://userver.ftw.at/~jossy/turbo/index.html>

- Encoding of a linear code is multiplication of an information block with a generator matrix.
- The goal is to produce a codeword which is longer than the information block and therefore contains redundancy.
- Every digit of the codeword is a linear combination of some digits of the information block.

- A simplification is to generate code digits only as linear combination of the last v information digits.
- If the combination is the same for every code digit we can implement it as a linear filter.
- The more parity sequences we generate, the more redundancy is added.

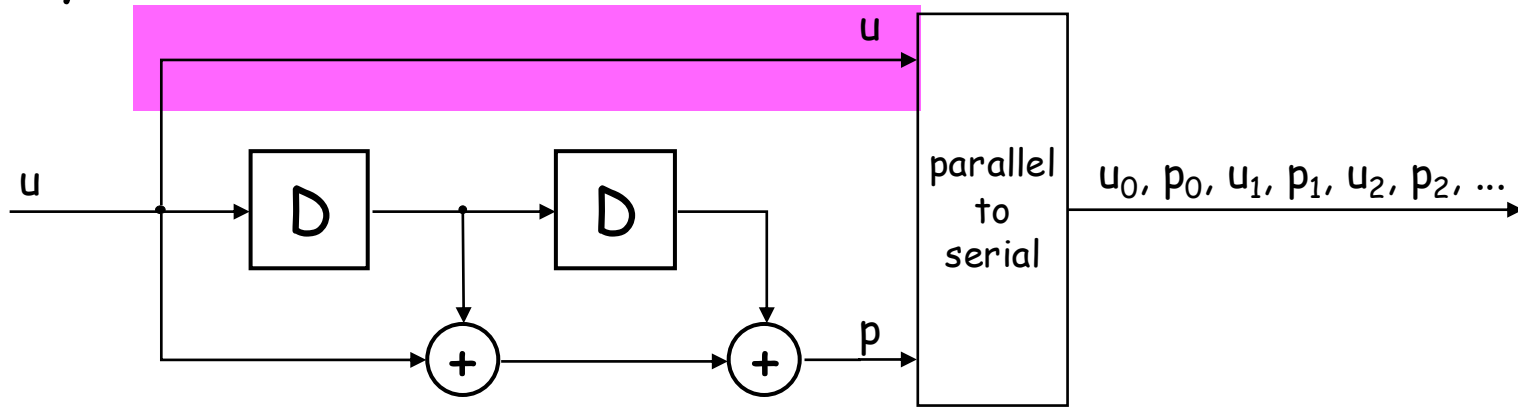




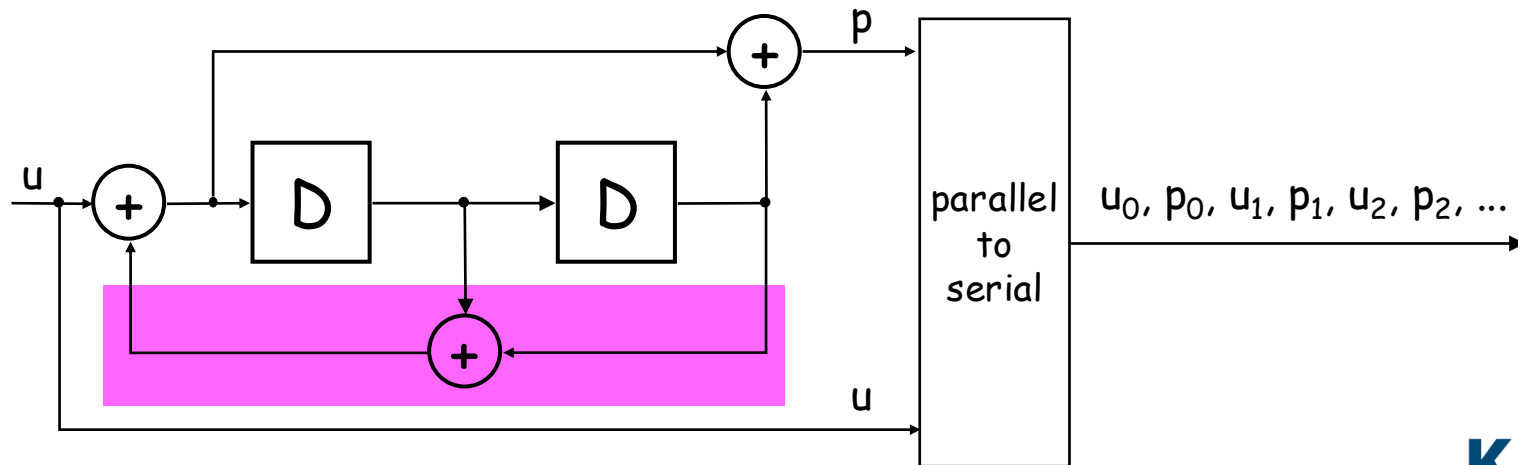
- The output sequence can be written as a convolution of the input sequence and the „impulse response“ of the filter

⇒ Convolutional Codes

Systematic encoder



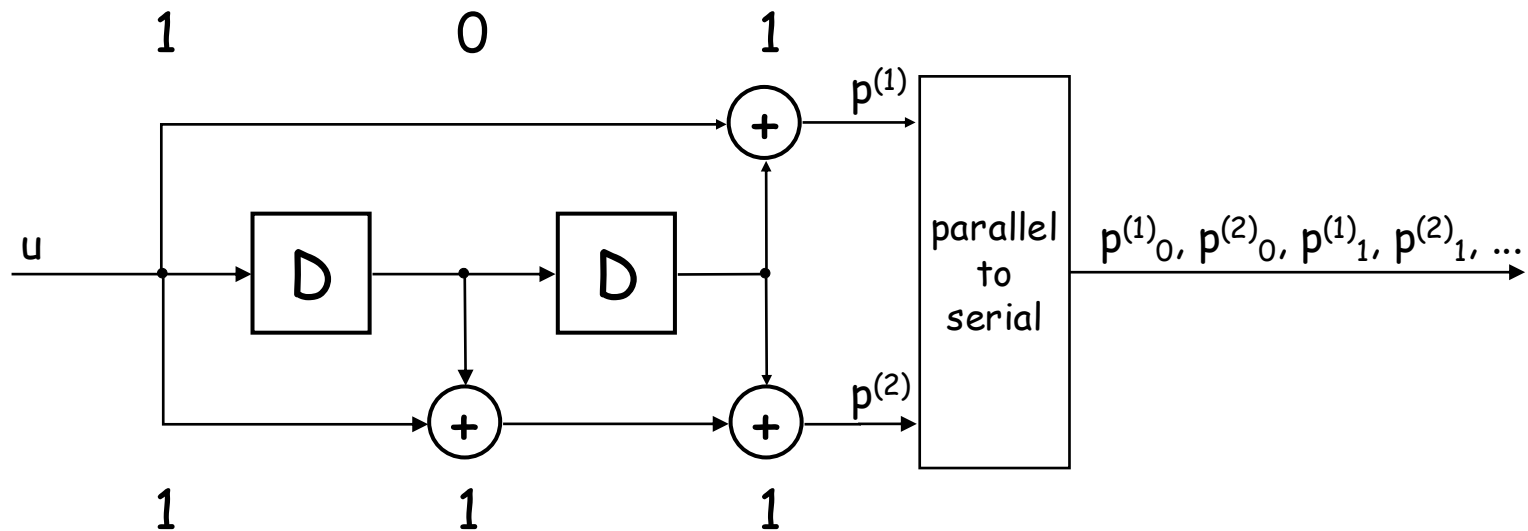
Recursive systematic encoder (RSC)



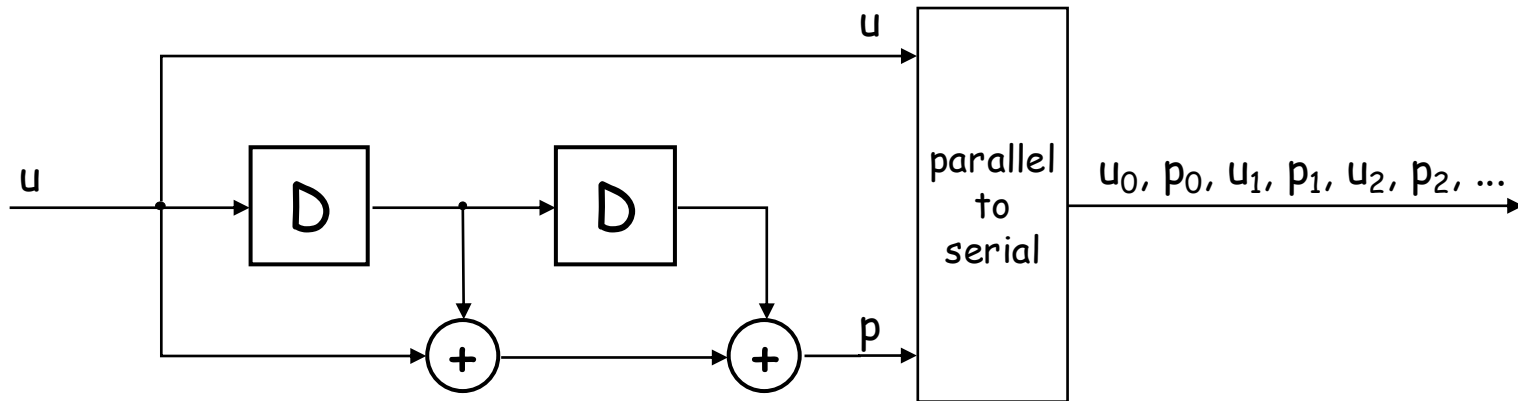
Notation

Octal numbers are used to describe the connections of the encoder.

The example below is written as 5/7 convolutional code.



Generator Matrix of a CC

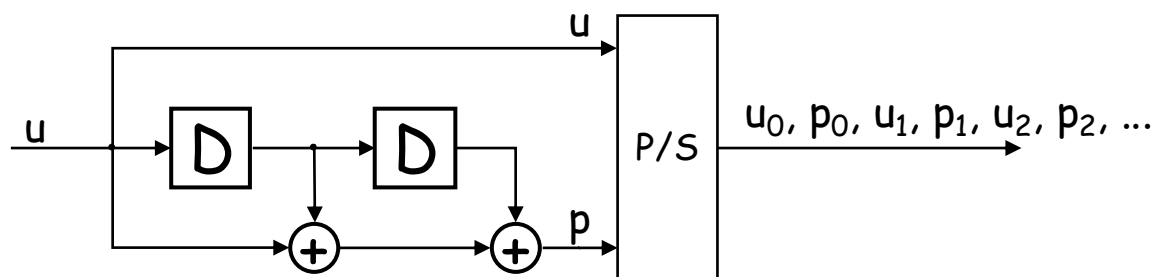


$$\begin{array}{c}
 \begin{array}{cccccccccccccccc}
 & u & p & u & p & u & p & u & p & u & p & u & p & u & p \\
 & 0 & 0 & 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 & 5 & 5 & 6 & 6
 \end{array} \\
 G = \begin{pmatrix}
 u_0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 u_1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 u_2 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
 u_3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
 u_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1
 \end{pmatrix}
 \end{array}$$

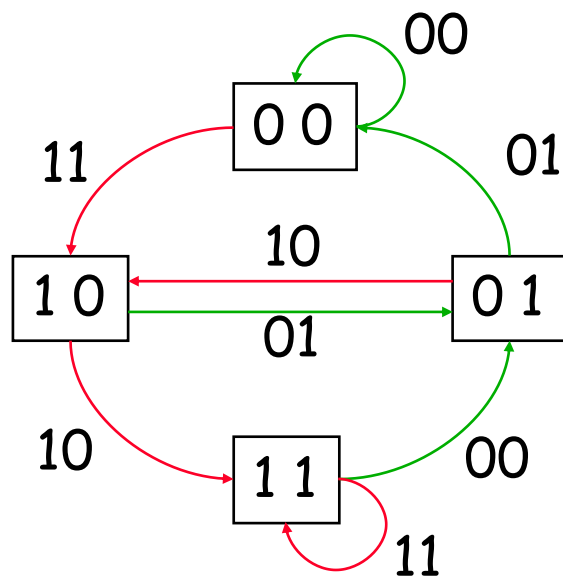
Band structure

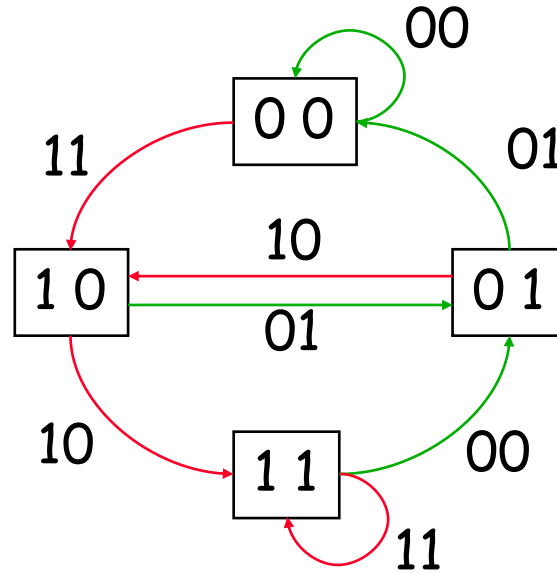
Output depends on the last v information symbols and the dependency is the same for every code symbol.

- At every timestep, the encoder is in a certain state which is simply described by the contents of its memory. It can be drawn as a state diagram:



input 0
input 1
output u p



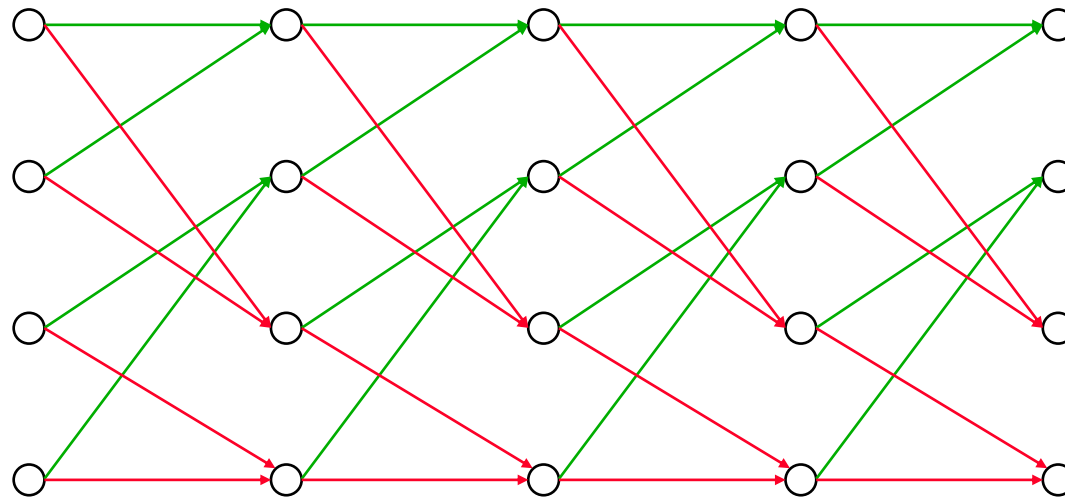


00

01

10

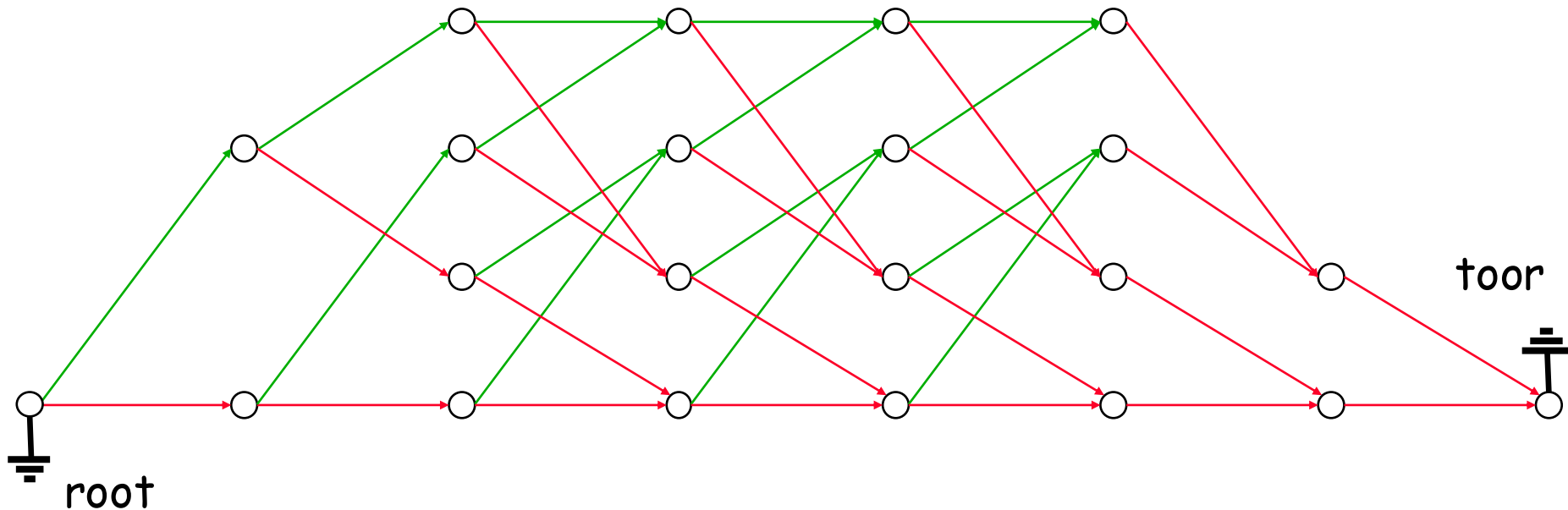
11



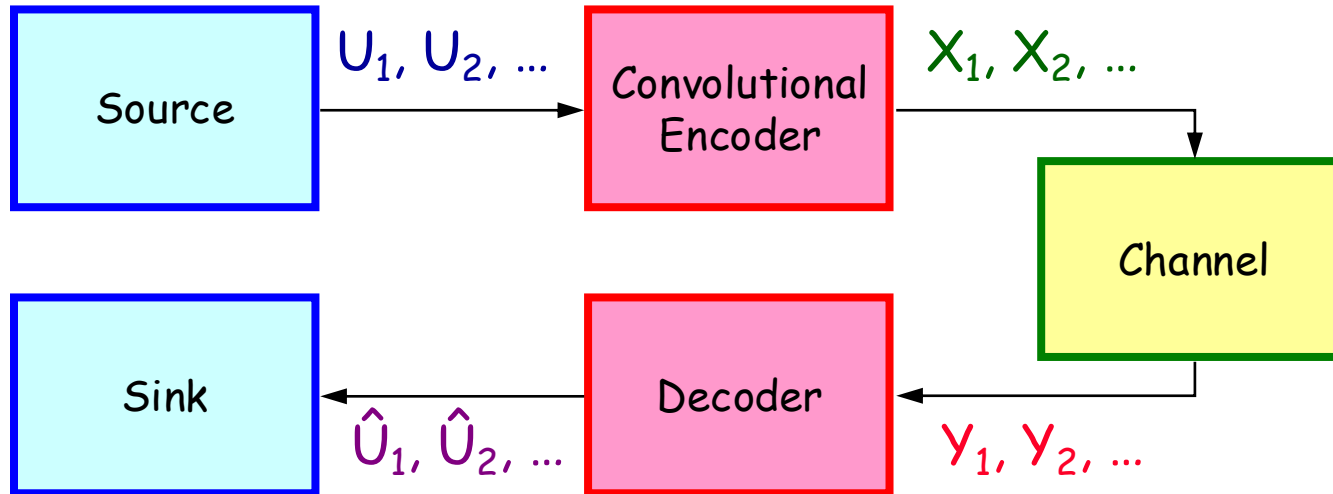
time index →

Terminated Trellis

- A convolutional encoder produces an **infinite** stream of data
- For decoding, we will consider the problem of decoding a **finite block of data**



Decoding



- Sequence decoding:

Choose the most likely u_1, \dots, u_K given the observation y_1, \dots, y_N

- Symbol decoding:

For all i , choose the most likely u_i given the observation y_1, \dots, y_N

Sequence Decoder

- General binary input memoryless channel:
 $P(y|X=0)$ and $P(y|X=1)$ given
- Sequence Decoder: $\max P(x_1 \dots x_N | y_1 \dots y_N)$ over all possible codewords $x = x_1 \dots x_N$, but

$$P(x_1 \dots x_N | y_1 \dots y_N) = P(y_1 \dots y_N | x_1 \dots x_N) P(x_1 \dots x_N) / P(y_1 \dots y_N),$$

$$P(x_1 \dots x_N) = 2^{-K}, \text{ and}$$

$$P(y_1 \dots y_N | x_1 \dots x_N) = \prod_i P(y_i | x_i)$$

Sequence decoder: for a received sequence $y = y_1 \dots y_N$,
 $\max \prod_i P(y_i | x_i)$ over all choices of codewords $x = x_1 \dots x_N$

Symbol Decoder

- Symbol decoder: for every i , choose x_i to maximize $P(x_i | y_1 \dots y_N)$, i.e.,

$$\begin{cases} x_i = 0 & \text{if } P(X_i=0 | y_1 \dots y_N) > P(X_i=1 | y_1 \dots y_N) \\ x_i = 1 & \text{if } P(X_i=0 | y_1 \dots y_N) < P(X_i=1 | y_1 \dots y_N) \\ x_i = 0 \text{ or } 1 & \text{if } P(X_i=0 | y_1 \dots y_N) = P(X_i=1 | y_1 \dots y_N) \end{cases}$$

or, in log-likelihood ratios: $x_i = \frac{1}{2} - \frac{1}{2} \text{sgn } L(X_i | y_1 \dots y_N)$

- But $P(X_i=0 | y_1 \dots y_N) = \sum_{x_i=0} P(x_1 \dots x_N | y_1 \dots y_N)$
and $P(x_1 \dots x_N | y_1 \dots y_N) = 2^{-K} \prod_j P(y_j | x_j) / P(y_1 \dots y_N)$

Symbol decoder: for a received sequence $y = y_1 \dots y_N$, for every i , choose $x_i = 0$ if $\sum_{x_i=0} \prod_j P(y_j | x_j) > \sum_{x_i=1} \prod_j P(y_j | x_j)$ and choose $x_i = 1$ otherwise

- Sequence decoder: compare 2^k products of N factors
- Symbol decoder: N sums of 2^k products of N factors

It is **impossible** to realize these decoders in practice for general codes

Sequence Decoding: the Viterbi Algorithm

Question: is $P(011011100011|y_1\dots y_N)$, $P(110101100011|y_1\dots y_N)$??

If two codewords have the same suffix,
 $x = x_1\dots x_k x_{k+1}\dots x_N$ and $z = z_1\dots z_k x_{k+1}\dots x_N$
then $P(x|y_1\dots y_N)$, $P(z|y_1\dots y_N)$ if and only
if $\prod_{i=1\dots k} P(y_i|x_i)$, $\prod_{i=1\dots k} P(y_i|z_i)$

For $k=1\dots N$, compute $\prod_{i=1\dots k} P(y_i|x_i)$ along every
path in the trellis. Whenever two paths meet,
discard the path with the lower probability!

2001年4月号特集

ednjapan.comment

21世紀の無線通信技術「世代」の過去と未来

アンドリュー・ビタビ
(Andrew J. Viterbi) 氏

符号化手法の最尤(さいゆう)判定法の1つであるビタビ・アルゴリズムを考案。米クアルコム社の元副会長。退任後、ビタビ・グループ(The Viterbi Group)を設立。クアルコム社はCDMA方式携帯電話の開発企業。



----> *用語の解説
(説明は本誌が作成)

「世代」という言葉で無線技術の進歩を分類することが一般化している。1980年代初期のアナログ携帯電話は、帯域幅が(1チャンネル当たり)30kHz以下に制限されており、第1世代と呼ばれている。この技術のブレイクスルーは、単に周波数変調(FM)の何10年間にもわたる進化の過程で出てきた伝送手段ということにあるのではなく、多数の基地局を設置して、これを各地域のユーザーに割り当てることによって、端末の電力を減らし、システム容量を何倍にも増やせるという「セルラー」*の概念にあった。

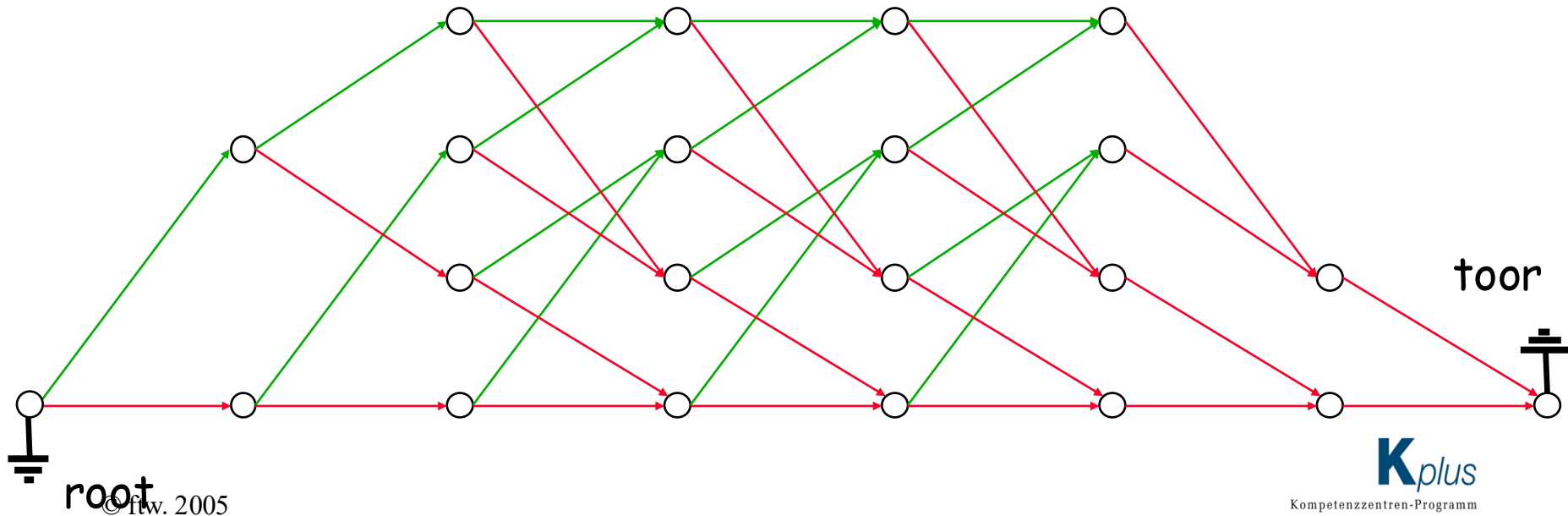
デジタル電話機は主に音声用として設計され、80年代後期から90年代初期に

One of the most famous and
Successful information theorists

Co-founded Link-a-bit and
Qualcomm

The Viterbi Algorithm

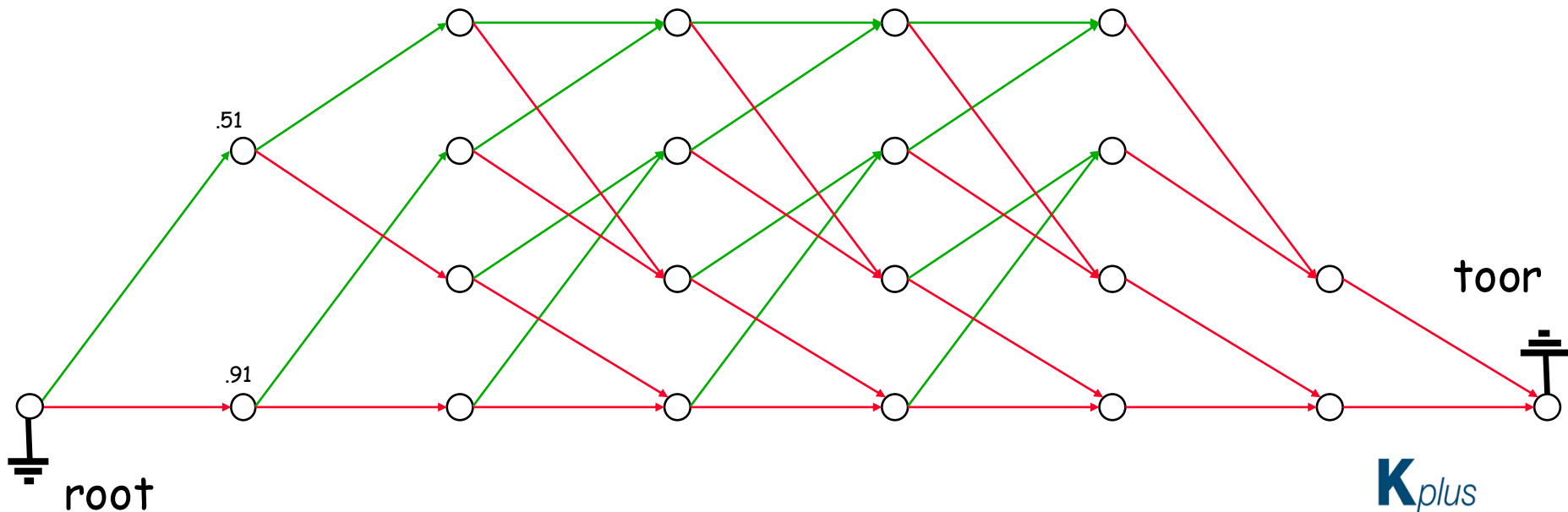
- In practice, it is easier (and equivalent) to minimize $\sum_i -\log P(y_i|x_i)$ than to maximize $\prod_i P(y_i|x_i)$
- **Exercise:** the log-likelihood ratios corresponding to the received word out of a BIAWGN channel are +0.4, -1.2, +0.2, +0.8, -0.3, -0.1, -0.3. What is the most likely transmitted codeword?



Exercise

- Convert LLRs to $-\log(P_{x|y}(0,y))$ and $-\log(P_{x|y}(1,y))$
- $-\log(P_{x|y}(0,y)) = \log(1+\exp(\text{LLR}))$
- $-\log(P_{x|y}(1,y)) = \log(1+\exp(\text{LLR})) - \text{LLR}$

LLR	.4	-1.2	.2	.8	-.3	-.1	-.3
$-\log(P_{x y}(0,y))$.91	.26	.80	1.17	.55	.64	.55
$-\log(P_{x y}(1,y))$.51	1.46	.60	.37	.85	.74	.85

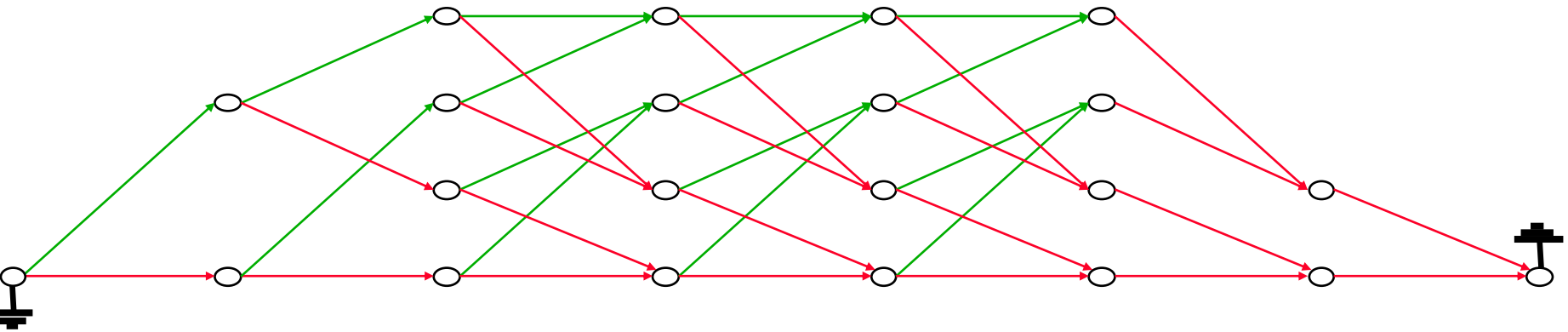


Symbol decoding: the BCJR Algorithm

(Bahl, Cocke, Jelinek & Raviv, 1974)

(also known as the forward-backward algorithm)

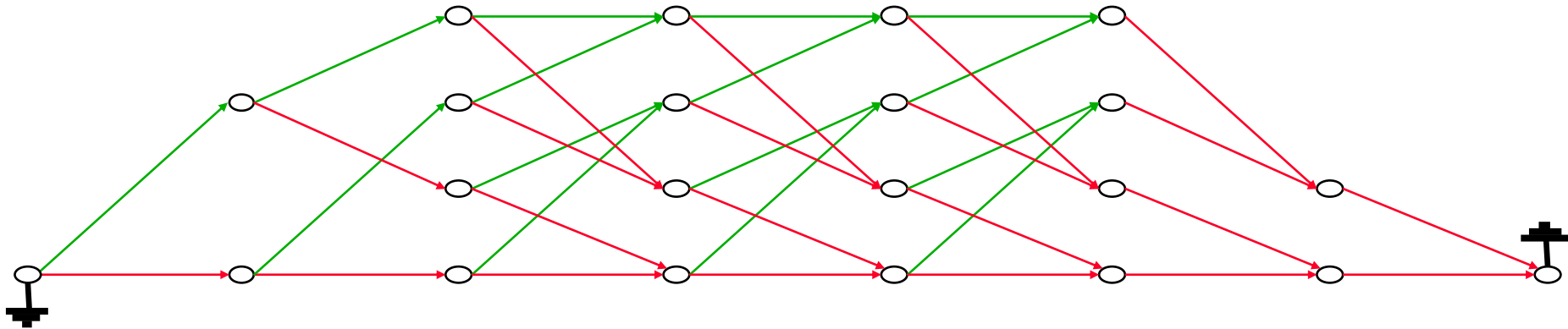
We want to calculate $\sum_{x_i=0} \prod_j P(y_j | x_j)$
and $\sum_{x_i=1} \prod_j P(y_j | x_j)$



We can separate the sums for X_i in sums over each red (or green) edge at depth i , i.e.,

$$\sum_{x_i=0} \prod_j P(y_j | x_j) = \sum_{\text{red edges at depth } i} \sum_{\text{paths through edge}} \prod_j P(y_j | x_j)$$

Symbol decoding: the BCJR Algorithm



We now write $\sum_{\text{paths through a red edge at depth } i} \prod_j P(y_j | x_j)$ as

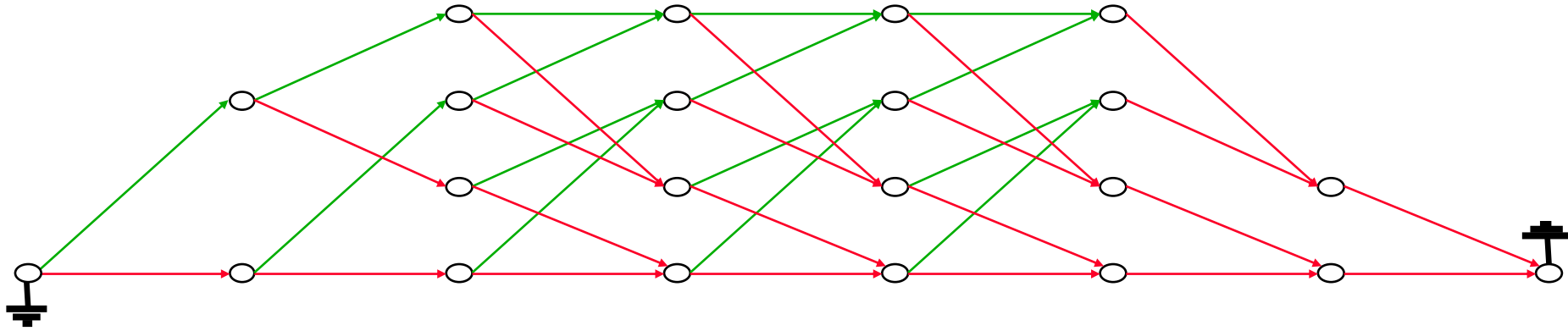
the product of:

$$P(y_i | X_i=0),$$

$$\sum_{\text{paths ending in origin state}} \prod_j P(y_j | x_j), \text{ and}$$

$$\sum_{\text{paths beginning in destination state}} \prod_j P(y_j | x_j),$$

since every combination of a path ending in the origin state of the edge, the edge itself and a path beginning in the destination state of the edge is a valid path (codeword) passing through our edge



The expression $\sum_{\text{paths ending in origin state}} \prod_j P(y_j | x_j)$ can be computed recursively by summing the paths ending at states to the left of our state and multiplying them by the $P(y_k | x_k)$ on the connecting edge

The same is true for $\sum_{\text{paths beginning in destination state}} \prod_j P(y_j | x_j)$ (except we sum recursively over the states to the right of our state)

The BCJR Algorithm

1.) For every edge, compute $\gamma_i = P(y_i | x_i)$
(where $x_i=0$ for a red edge and $x_i=1$ for a green edge)

2.) For the root, define $\alpha_{\text{root}} = 1$

Starting from the root, for every node n , compute

$$\alpha_n = \sum_{n' \text{ left of } n} \gamma_{n' \rightarrow n} \alpha_{n'}$$

3.) For the toor, define $\beta_{\text{toor}} = 1$

Starting from the toor, for every node n , compute

$$\beta_n = \sum_{n' \text{ right of } n} \gamma_{n \rightarrow n'} \beta_{n'}$$

4.) For every edge, compute $\alpha_{\text{origin}} \gamma \beta_{\text{destination}}$

5.) At every depth, sum the values on the red edges

to obtain $\sum_{x_i=0} \prod_j P(y_j | x_j)$ and the values on the

green edges for $\sum_{x_i=1} \prod_j P(y_j | x_j)$